GE VERNOVA

EDGE SOFTWARE & SERVICES

EDGE AGENT

User Guide

GE VERNOVA

Proprietary Notice
The information contained in this publication is believed to be accurate and reliable. However, GE Vernova assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

Trademark Notices
"VERNOVA" is a registered trademark of GE Vernova. "GE VERNOVA" is a registered trademark of GE Aerospace exclusively licensed to GE Vernova. The terms "GE" and the GE Monogram are trademarks of GE Aerospace and are used with permission.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:
doc@ge.com

Contents

Edge Core API

Edge Core API

The Edge Core API allows users to interact with the system utilizing the same underlying functionality as Edge Manager or the Predix Technician Edge Console (PETC). The API is accessible from inside the device when logged into a terminal via a Unix domain socket. The API can be used for development purposes to manage applications, configure the network, perform host updates, retrieve logs, etc.

The API is not accessible from outside the device and is subject to change in future releases.

Applications

Info

Gets details about the specified application.

Command: GET

```
http://localhost/api/v1/applications/<app_name>
```

Curl example

```
curl http://localhost/api/v1/applications/my-app \
    --unix-socket /var/run/edge-core/edge-core.sock
```

Success response example

```
{
    "container_app_info": {
        "application": {
            "name": "my-app",
            "service": [
                {
                    "current_state": "Running 24 hours ago",
                    "desired_state": "Running",
                    "error_message": "",
                    "id": "fuz7x5vjru8w1li11fhgcixuu",
                    "image": "eh-server",
                    "name": "my-app_app2.1"
                },
```

```
                {
                    "current_state": "Running 24 hours ago",
                    "desired_state": "Running",
                    "error_message": "",
                    "id": "aiznk23c3mfpi7ltbk54hx5yp",
                    "image": "ds-server",
                    "name": "my-app_app1.1"
                }
            ],
            "version": ""
        }
    },
    "platform_name": "Docker",
    "platform_version": "Docker version 17.05.0-ce, build 89658be"
}
```

Unsuccessful response example

```
{
    "error_message": "application not found: my-appx",
    "status_code": 400
}
```

Delete

Deletes a deployed application.

Command: DELETE

```
http://localhost/api/v1/applications/<app_name>
```

Table 1. Paramater

Field	Type	Description
app_name	String	Application name

Curl example

```
curl http://localhost/api/v1/applications/my-app \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X DELETE
```

Table 2. Error 4xx

Name	Type	Description
error_message	String	A description of the error encountered
status_code	Number	Status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Deploy Configuration

Deploys the configuration of an application.

Command: POST

```
http://localhost/api/v1/applications/<app_name>/configuration
```

Table 3. Parameters

Field	Type	Description
app_name	String	Application name
file	String	Application configuration file

Curl example

```
curl http://localhost/api/v1/applications/my-app/configuration \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X POST \
    -F "file=@/mnt/data/downloads/my-app-config.zip"
```

Success response example

```
{
    "app_config_deploy": "App configuration deployment successful."
}
```

Error response example

```
{
  "error_message": "Application with id my-app does not exist.
   Unable to deploy configs.",
  "status_code": 400
}
```

Deploy Application

Deploys an application.

Command: POST

```
http://localhost/api/v1/applications
```

Table 4. Parameters

Field	Type	Description
app_name	String	Application name
file_name	String	Name of the application deployment file

Curl example

```
curl http://localhost/api/v1/applications \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X POST \
    -F "file=@<file_name>" \
    -H "app_name: my_app"
```

Success response example

```
{
    "app_deploy": "Application deployment successful."
}
```

Error response example

```
{
  "error_message": "gzip: stdin: unexpected end of file tar:
   Child returned status 1 tar: Error is not recoverable:
   exiting now",
```

```
  "status_code": 400
}
```

Get List

Gets a list of applications.

Command: GET

```
http://localhost/api/v1/applications
```

Curl example

```
curl http://localhost/api/v1/applications \
    --unix-socket /var/run/edge-core/edge-core.sock
```

Success response example

```
{
    "applications": [
        "my-app",
        "web-server",
        "database"
    ]
}
```

Table 5. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	Status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Start an Application

Starts an application.

Command: POST

```
http://localhost/api/v1/applications/<app_name>/start
```

Table 6. Parameter

Field	Type	Description
app_name	String	Application name

Curl example

```
curl http://localhost/api/v1/applications/my-app/start \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X POST
```

Table 7. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	Status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Stop an Application

Stops an application.

Command: POST

```
http://localhost/api/v1/applications/<app_name>/stop
```

Table 8. Parameter

Field	Type	Description
app_name	String	Application name

Curl example

```
curl http://localhost/api/v1/applications/my-app/stop \

    --unix-socket /var/run/edge-core/edge-core.sock \

    -X POST
```

Table 9. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	Status code for the HTTP request

Error response example

```
{

    "error_message": "<errorMessage>",

    "status_code": 400

}
```

Device

Get OS Info

Gets the OS information of the device.

Command: GET

```
http://localhost/api/v1/host/os
```

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/os
```

Table 10. Success 200

Field	Type	Description
os_name	String	OS name
os_version	String	OS version
os_arch	String	OS architecture

Success response example

```
{
    "os_name": "Edge OS",
    "os_version": "2.0.0-beta.8",
    "os_arch": "x86_64"
}
```

Get Update State

Gets the state of an update.

State is kept for the most recent update attempt, so querying with an older update_identifier will return “NO UPGRADE OCCURRED”. Note: update_identifier is required because there is a brief lag between the start of an update and the actual upgrade process running. So if a client starts an update and then immediately tries to get the ‘state’ before the upgrade process has started, without the id they would get the state of the previous update, an incorrect result.

Command: GET

```
http://localhost/api/v1/host/state?update_identifier=<identifier>
```

Table 11. Parameter

Field	Type	Description
update_identifier	String	Identifier of the update

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \
    http://localhost/api/v1/host/state?update_identifier=<identifier>
```

Table 12. Success 200

Field	Type	Description
last_os_upgrade	String	State of the update associated with the identifier: "PASS", "FAIL", INPROGRESS" or "NO UPGRADE OCCURRED"

Success response example

```
{
    "last_os_upgrade": "PASS"
}
```

Reboot

Reboots the device.

Command: POST

```
http://localhost/api/v1/host/reboot
```

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \
     -X POST http://localhost/api/v1/host/reboot
```

Upon successful completion of the operation, the box will reboot.

Update

Updates the Edge OS and Edge Agent.

Command: POST

```
http://localhost/api/v1/host/update
```

Table 13. Parameters

Field	Type	Description
update_identifier	String	An identifier associated with the update. Can be any string, as long as it is unique.

Table 13. Parameters (continued)

Field	Type	Description
file_name	String	Name of the Edge OS/Edge Agent update image file.

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     -X POST -H "update_identifier: <identifier>" -F "file=@<file_name>" \

     http://localhost/api/v1/host/update
```

If the update is successfully applied, the device will reboot to complete the update.

Table 14. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{

    "error_message": "<errorMessage>",

    "status_code": 500

}
```

Enrollment

Delete

Delete enrollment information for an enrolled device.

Command: DELETE

```
http://localhost/api/v1/host/enroll/
```

Curl example

```
curl http://localhost/api/v1/host/enroll \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X DELETE
```

Table 15. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	Status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Enroll Device

Enrolls the device with the Edge Manager.

Command: POST

```
http://localhost/api/v1/host/enroll
```

Table 16. Parameters

Field	Type	Description
sharedSecret	String	Shared secret
deviceId	String	Device identifier
enrollmentUrl	String	URL for the enrollment

Curl example

```
curl http://localhost/api/v1/host/enroll \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X POST \
    -H "Content-Type: application/json" \
```

```
    -d '{
            "sharedSecret": "secret",
            "deviceId": "some-device-id",
            "enrollmentUrl": "https://some-enrollment-url"
        }'
```

Table 17. Success 200

Field	Type	Description
enrollment	String	"Certified enrollment success"

Success response example

```
{
    "enrollment": "Certificate enrollment success"
}
```

Table 18. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Get Enrollment Status

Gets enrollment details from the specified device.

Command: GET

```
http://localhost/api/v1/host/enroll
```

Curl example

```
curl http://localhost/api/v1/host/enroll \

    --unix-socket /var/run/edge-core/edge-core.sock
```

Table 19. Success 200

Field	Type	Description
enrolled	Boolean	True if device is enrolled
deviceId	String	Device identifier
edgeManagerUrl	String	URL for the device manager

Enrolled device example

```
{

    "enrolled": true,

    "deviceId": "some-device-id",

    "edgeManagerUrl": "https://edgemanager-sysint.predix.io"

}
```

Unenrolled device example

```
{

    "enrolled": false

}
```

Logs

Get Log

Use journalctl to retrieve logs. The parameter descriptions reference sections in man journalctrl.

Command: GET

```
http://localhost/api/v1/host/logs
```

Table 20. Parameters

Field	Type	Description
unit	String	Filter logs by the given UNIT (see --unit).

Table 20. Parameters (continued)

Field	Type	Description
priority	String	Filter the logs by priority (See --priority).
useJSON	Boolean	Format output as JSON (See --output=json).
useUTC	Boolean	Format timestamps in UTC time (See --utc).
since	String	Filter logs before DATE (See --since).
until	String	Filter logs after DATE (See --until).
kernel	Boolean	Filter logs; only show kernel (See --kernel).
boot	String	Filter out logs before the given boot (See --boot).
applicationService	String	Filter logs on APPLICATION SERVICE.
lines	Integer	Number of log lines to print.
reverse	Boolean	Print the newest entries first.

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/logs?since=1+hour+ago&useUTC=true
```

Table 21. Success 200

Field	Type	Description
log_text	String	Log content

Success response example

```
{

    "log_text": "Nov 28 11:04:40 machine-name kernel: sd 2:0:0:0: [sda] \
```

```
                Assuming drive cache: write through\\n"
}
```

Table 22. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Network

Enable DHCP

Enables DHCP on a network interface.

Command: PUT

```
http://localhost/api/v1/host/network/interfaces/<interface>/dhcp
```

Table 23. Parameter

Field	Type	Description
interface	String	Interface name

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \
     -X PUT http://localhost/api/v1/host/network/interfaces/enp0s3/dhcp
```

Table 24. Success 200

Field	Type	Description
dhcp	Boolean	Indicates if DHCP is used or not
name	String	Interface name
type	String	Interface type

Success response example

```
{
    "dhcp": true,
    "name": "enp0s3",
    "type": "ethernet"
}
```

Table 25. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "Invalid Interface",
    "status_code": 400
}
```

Get NTP Information

Gets NTP information.

Command: GET

```
http://localhost/api/v1/host/network/ntp
```

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/network/ntp
```

Table 26. Success 200

Field	Type	Description
server	String	NTP server IP address

Success response example

```
{
    "servers": [
        "192.168.233.228",
        "ntp.ubuntu.com",
        "time1.google.com"
    ]
}
```

Get Network Interface

Gets information for a specific network interface.

Command: GET

```
http://localhost/api/v1/host/network/interfaces/<interface>
```

Table 27. Parameter

Field	Type	Description
interface	String	Name of the interface

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/network/interfaces/enp0s3
```

Table 28. Success 200

Field	Type	Description
dhcp	Boolean	Indicates if DHCP is used or not

Table 28. Success 200 (continued)

Field	Type	Description
dns	String	DNS server IP address
gateway	String	Gateway IP address
ipv4	String	Interface IP address
name	String	Interface name
type	String	Interface type

Success response example

```
{
    "dhcp": true,
    "dns": "10.0.2.3,8.8.8.8",
    "gateway": "10.0.2.2",
    "ipv4": "10.0.2.15/24",
    "name": "enp0s3",
    "type": "ethernet"
}
```

Table 29. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Get Network Interfaces

Gets information for the device's network interfaces.

Command: GET

```
http://localhost/api/v1/host/network/interfaces
```

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/network/interfaces
```

Table 30. Success 200

Field	Type	Description
interfaces	Object	List of interfaces
dhcp	Boolean	Indicates if DHCP is used or not
dns	String	DNS server IP address
gateway	String	Gateway IP address
ipv4	String	Interface IP address
name	String	Interface name
type	String	Interface type

Success response example

```
{
    "interfaces": [
        {
            "dhcp": true,
            "dns": "10.0.2.3,8.8.8.8",
            "gateway": "10.0.2.2",
            "ipv4": "10.0.2.15/24",
            "name": "enp0s3",
            "type": "ethernet"
        }
    ]
}
```

Get Proxy Information

Gets network proxy information.

Command: GET

```
http://localhost/api/v1/host/network/proxy
```

Curl example

```
curl --unix-socket /var/run/edge-core/edge-core.sock \

     http://localhost/api/v1/host/network/proxy
```

Table 31. Success 200

Field	Type	Description
http	String	HTTP proxy IP address
https	String	HTTPS proxy IP address
no_proxy	String	Non-proxy servers

Success response example

```
{

    "http": "http://3.3.3.3:80",

    "https": "http://5.5.5.5:88",

    "no_proxy": "*.ge.com, *.google.com"

}
```

Set NTP Information

Sets the NTP server address.

Command: PUT

```
http://localhost/api/v1/host/network/ntp
```

Table 32. Parameter

Field	Type	Description
server	String	IP address of an NTP server

Curl example

```
curl http://localhost/api/v1/host/network/ntp \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X PUT \
    -H "Content-Type: application/json" \
    -d '{
          "servers" : ["192.168.233.228", "ntp.ubuntu.com", "time1.google.com"]
        }'
```

Table 33. Success 200

Field	Type	Description
server	String	IP address of an NTP server

Success response example

```
{
    "servers": [
        "192.168.233.228",
        "ntp.ubuntu.com",
        "time1.google.com"
    ]
}
```

Table 34. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Set Network Interface

Manually configures a network interface.

Command: PUT

```
http://localhost/api/v1/host/network/interfaces/<interface>/manual
```

Table 35. Parameters

Field	Type	Description
ipv4	String	Network interface IP address
gateway	String	Gateway IP address
dns	String	DNS server IP address

Curl example

```
curl http://localhost/api/v1/host/network/interfaces/enp0s3/manual \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X PUT \
    -H "Content-Type: application/json" \
    -d '{
            "ipv4": "192.168.233.228/24",
            "gateway": "192.168.233.2",
            "dns": "192.168.233.2"
        }'
```

Table 36. Success 200

Field	Type	Description
dhcp	Boolean	Indicates if DHCP is used or not
dns	String	DNS server IP address
gateway	String	Gateway server IP address
ipv4	String	IP network interface address
name	String	Network interface name
type	String	Network interface type

Table 37. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Set Proxy Information

Sets network proxy information. A reboot is required for the new proxy setting(s) to take effect. All running Edge applications will be redeployed automatically during the next reboot to pickup the new proxy values.

Command: PUT

```
http://localhost/api/v1/host/network/proxy
```

Table 38. Parameters

Field	Type	Description
http	String	HTTP proxy IP address
https	String	HTTPS proxy IP address
no_proxy	String	Non-proxy servers

Curl example

```
curl http://localhost/api/v1/host/network/proxy \
    --unix-socket /var/run/edge-core/edge-core.sock \
    -X PUT \
    -H "Content-Type: application/json" \
    -d '{
            "http" : "http://192.168.223.228:80",
            "https" : "http://192.168.233.229:80",
```

```
            "no_proxy": "google.com"

        }'
```

Table 39. Success 200

Field	Type	Description
proxy	String	"Proxy set successfully"

Success response example

```
{
    "proxy": "Proxy set successfully"
}
```

Table 40. Error 4xx

Field	Type	Description
error_message	String	A description of the error encountered
status_code	Number	The status code for the HTTP request

Error response example

```
{
    "error_message": "<errorMessage>",
    "status_code": 400
}
```

Edge Agent on Ubuntu

Installation

Edge Agent on Ubuntu is distributed as a debian package, similar to many other packages on Ubuntu. The most popular tool for installing debian packages is `apt`. The following sections provide instructions for installing Edge Agent from a downloaded `.deb` file from the public apt repository on dig-grid-artifactory.apps.ge.com, and for setting up your own apt repository and updating Edge Agent from there.

Note:
`apt` is a large and complex tool built on top of `dpkg`, which is itself a complex program with many configuration options and security considerations. The following information is not intended to be exhaustive, and users should read the documentation for `apt`, `dpkg`, and associated tools. Relevant manual pages include (but are not limited to): `apt`, `dpkg`, and `sources.list`. Run man <page> in your terminal to view an entry.

Note:
Before installing Edge Agent, ensure your Ubuntu system is up-to-date by running the following commands:

```
sudo apt update

sudo apt upgrade
```

Install From Downloaded Package

Download the Debian package from https://dig-grid-artifactory.apps.ge.com/ui/native/predix-edge-agent-deb/pool/ to your device running Ubuntu. Run the command `apt install /path/to/predix-edge-agent.deb` (substituting the actual path to the downloaded `.deb` file) as root to install it. Edge Agent should now be ready for enrollment with Edge Manager.

Note:
Please be sure to download the latest version with the appropriate architecture for your system.

Note:
To update Edge Agent, download the newer `.deb` file and follow these instructions again.

Accessing Artifactory Downloads

To access Artifactory downloads, you will require a GE SSO (single sign-on) username and approval to access Artifactory.

Request a GE SSO

Use the following steps to obtain a GE SSO if you do not already have one.

1. Complete the Your GE SSO Account request form. All fields marked with a checkmark are required.
2. Click **Submit**.

> **Note:**
> The only non-alphanumeric characters allowed in your GE SSO are an underscore (_) and a period (.). Using any other non-alphanumeric characters for your username will result in an invalid authentication in Artifactory.

Request Artifactory Access

Once you have an SSO, use the following steps to request access to Artifactory.

1. Complete the Edge Artifactory Access Requests form.
2. Click **Submit**.

Setup Edge Agent

1. If required in your environment, set a proxy in `/etc/environment`.
    - Set `http_proxy` and `https_proxy` to the internet address and port of your proxy. Also, set the value for `no_proxy` for addresses that should not go through the proxy. For example:

      ```
      http_proxy=http://10.9.8.7:6543
      https_proxy=http://10.9.8.7:6543
      no_proxy=localhost,127.0.0.1,mycompany.com
      ```

    - Reboot or re-login to apply proxy variables.
2. (Optional) Create a device model.
    - You can choose to create a custom device model on Edge Manager, which can be useful for organizing your inventory of devices. Instructions for doing this can be found here
3. Enroll your device.

- Create your device on Edge Manager. Note the device ID, secret, and URL of your Edge Manager instance as this information will be required later.
- Ensure the system time is correct. Edge-to-cloud communications will fail if your Edge device and your Edge Manager instance are not at least relatively close in time.
- Run the Helper Script *(on page xxx)* as "root".
  - The script will notify on success or failure of the operation.
  - For more information, see Enrolling a device.

Edge Agent should begin communicating with Edge Manager, which should show your device as **Online** after a short wait. This can be verified on Edge Manager by navigating to **Device Manager > Devices** and searching for your new device.

Installation Helper Script

The following script automates enrolment of Edge Agent on Debian. It should be run as root so that it has permission to run Edge Agent processes as the eauser and eagateway users.

Note:
The script does not modify the system time; enrollment may fail if the clock is not set correctly.

```
#!/bin/bash


set -e


ENROLL_INPUT="/var/lib/edge-agent/config/enrollment-input.json"

ENROLL_OUTPUT="/var/lib/edge-agent/config/enrollment-output.json"


if [ ! -f $ENROLL_OUTPUT ]; then

   if [ ! -f $ENROLL_INPUT ] || jq -e 'any(.[]; . == "")' $ENROLL_INPUT; then

      printf "First, make sure you have created your device on Edge Manager, "

      printf "and note the 'Device ID' and 'Secret' you used.\n\n"


      read -p "Enter the enrollment URL:" url

      read -p "Enter the device ID:" ID

      read -p "Enter the device secret:" secret


      url="$url" ID="$ID" secret="$secret" jq -n \
```

```
        '{deviceId: env.ID, sharedSecret: env.secret, enrollmentUrl: env.url}' \
        > $ENROLL_INPUT
   fi


   echo "Attempting device enrollment..."
   if ! su eauser -s /bin/bash -c "/opt/edge-agent/edge-agent-enrollment $ENROLL_INPUT"; then
      echo "Enrollment failed"
      exit 1
   fi
   echo "Enrollment succeeded"
fi


echo "Edge Agent is enrolled"
```

Uninstall Edge Agent

To remove Edge Agent, run the following command as root:

```
dpkg --remove --force-remove-essential predix-edge-agent
```

Usage

Limitations for Docker Compose

The docker compose file cannot contain volumes with these reserved targets:

- `/config`
- `/data`
- `/edge-agent`
- `/edge-core`
- `/shared`

If the image is to be pulled from the DTR, it should be referenced in the docker compose file with the digest in the format `dtr/image@digest`. Get the digest for current images by running docker `images --digests`.

If the image is to be loaded from a tar file, it should be saved and referenced in the docker compose file without the DTR prefix in the format `image:tag`.

Application Builder Helper Script

The following script (`edge-app-build`) is provided to help package a Docker app into the expected Edge App `tar.gz` archive format.

```
% /usr/bin/python3 ./edge-app-build --help
```

Usage: `edge-app-build [-h] [-n APP_NAME] [-b BUILD_DIR] [-o OUTPUT_DIR] [-f] src`

Table 41. Required Arguments

Argument	Description
src	The path to the folder containing the `docker-compose.yml` and other files for this application.

Table 42. Optional Arguments

Argument	Description
-h, --help	Show this help message and exit.
-n APP_NAME, --app-name APP_NAME	The name of the application (default = prompt()).
-b BUILD_DIR, --build-dir BUILD_DIR	The directory to create the Edge application in.
-o OUTPUT_DIR, --output-dir OUTPUT_DIR	The directory to save the built Edge application in.
-f, --force	Overwrite the build and output directories without prompting.

Docker Login/Logout

If your Edge Applications will pull images from a private docker registry, you need to create a custom command on Edge Manager to execute the "Docker Login" command on your Edge devices so that Edge Agent can authenticate with the docker registry. You create this command with Edge Manager's Add Command *(on page)* feature. You may want to also create a docker logout command in case you want to prevent an Edge Device from pulling from the registry in the future.

Docker Login

Use the following field values to create the Docker Login command:

- **Display Name**: Docker Login
- **Command**: docker-login
- **Handler**: ApplicationManager
- **Has Output**: <checked>
- **Platform**: Predix Edge
- **Parameters**:
    1. **Parameters Name**: registry

       Key: registry

       Value: <blank or desired default>

       Editable: <checked>
    2. **Parameters Name**: username

       Key: username

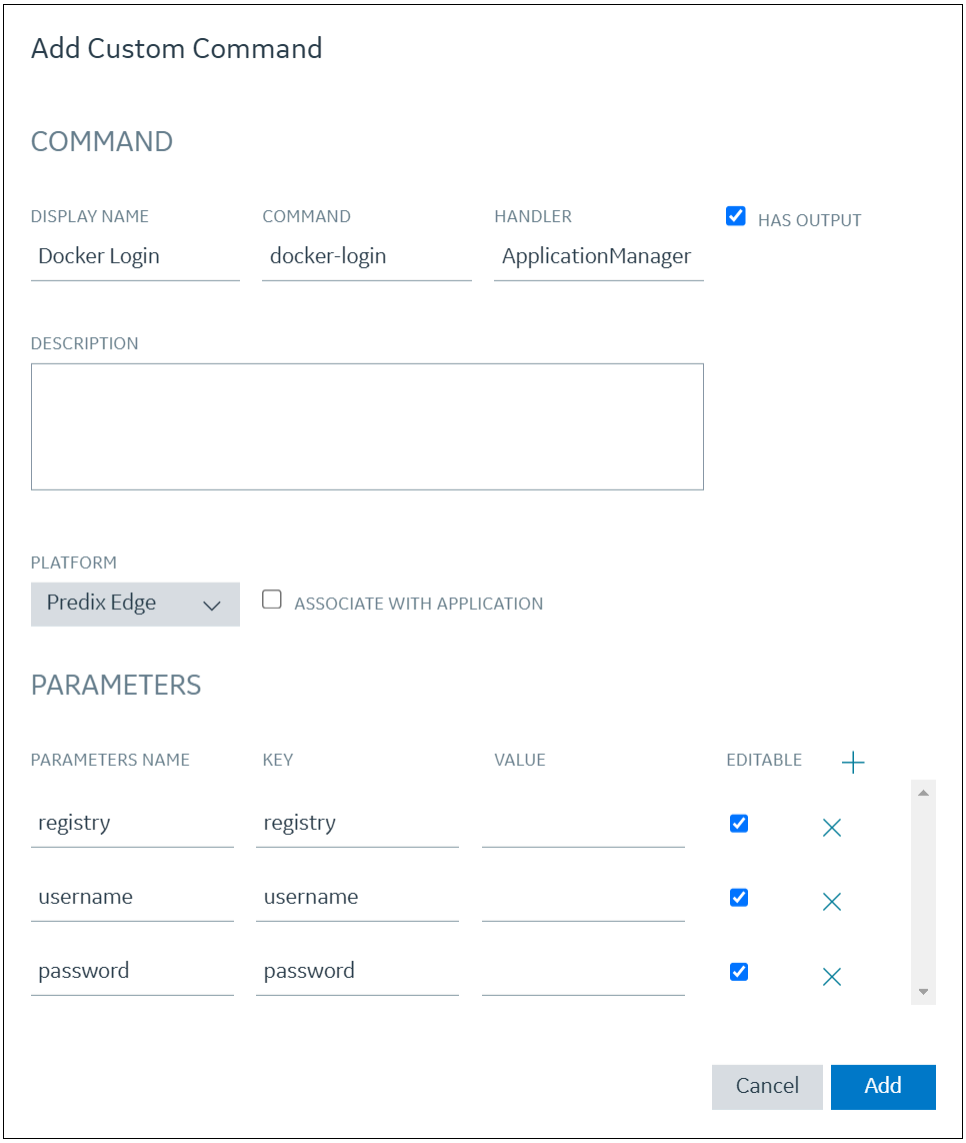
       Value: <blank or desired default>

       Editable: <checked>
    3. **Parameters Name**: password

       Key: password

       Value: <blank or desired default>

       Editable: <checked>



Docker Logout

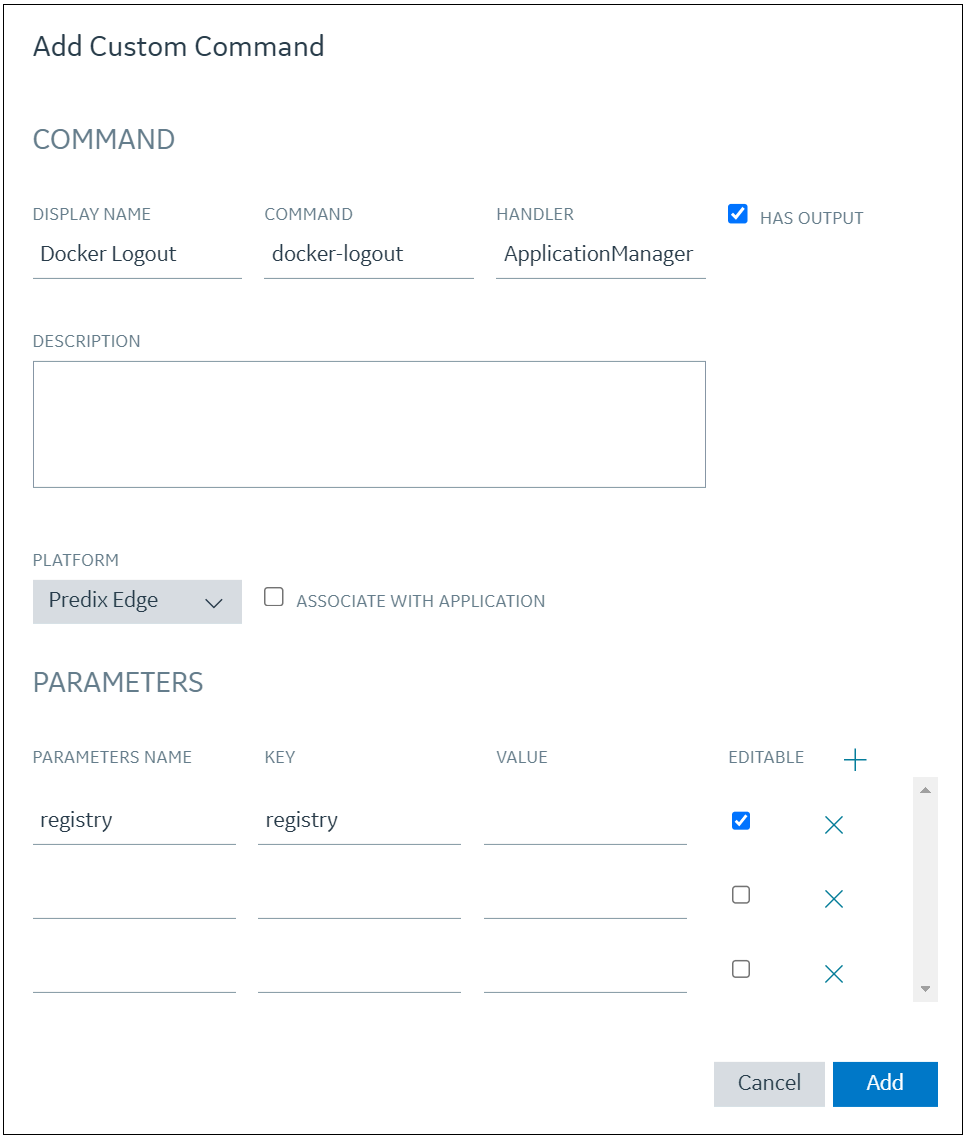
Use the following field values to create the Docker Login command:

- **Display Name**: Docker Logout
- **Command**: docker-logout
- Handler: ApplicationManager
- Has Output: <checked>
- **Platform**: Predix Edge
- **Parameters**:
    1. **Parameters Name**: registry

       Key: registry

Value: <blank or desired default>

Editable: <checked>



System Builder Commands

Introduction

This section describes some of the specific requirements for adding system builder commands to the Edge Agent.

Handler Files

New commands can be added by including new `*.json` files in the `handler_configs` directory. This directory is specified in the `paths.handler_configs` configuration item in the main Edge Agent configuration file (`/etc/edge-agent/agent-data.json`, points to this directory). Typically, this directory is `/etc/edge-agent/handler-configs`.

A handler consists of the following fields:

Table 43.

Field	Required	Description
name	Yes	Name of the handler
capabilities	No	Array of capabilities provided by the handler
path	Yes	Location of the handler's commands
commands		List of commands supported by the handler

A command consists of the following fields:

Table 44.

Field	Required	Description
bin	Yes	Location of the executable file on the system
needs_task_id	No	If set to `true` this will pass a task ID to the command. Defaults to `false` if omitted. For more infor-

Table 44. (continued)

Field	Required	Description
		mation see Writing Package Deployment Commands *(on page xl)*.

The format of the file is:

```
{
    "handlers": {
        "Machine": {
            "capabilities": [
                {
                    "name": "SomeCapability",
                    "version": "1.0"
                }
            ],
            "path": "/opt/something/machine_handlers",
            "commands": {
                "some-command": {
                    "bin": "some-exe",
                    "needs_task_id": true
                }
            }
        },
        "Host": {
            "capabilities": [
                {
                    "name": "Dir",
                    "version": "0.1"
                },
                {
                    "name": "Date",
                    "version": "1.1"
                }
            ],
            "path": "/bin",
            "commands": {
```

```
                "ls": {
                    "bin": "ls"
                },
                "date": {
                    "bin": "date"
                }
            }
        }
    }
}
```

In this example, "Machine" and "Host" are the handler names.

"Machine" provides version 1.0 of the `SomeCapability` capability. It supports a single command, `some-command`, that is implemented by `some-exe`, which is given a task ID. It expects these commands to be in `/opt/something/machine_handlers`.

"Host" provides version 0.1 of the `Dir` capability and version 1.1 of the `Date` capability. It supports two commands, `ls` and `date`. They are implemented by `ls` and `date` respectively. It expects these commands to be in `/bin`.

Capabilities are defined in the glossary.

Loading Handler Files

All `*.json` files in the handler configuration directory are loaded in lexical order. If a handler of a specified name has already been loaded and a handler of the same name is specified in a later configuration file, the new file is skipped, an error is logged, and the next file (if present) is processed.

Handlers are loaded only once, when the dispatcher is started. In order to reload the configuration files, the dispatcher (e.g., systemctl restart edge-agent-dispatcher) should be restarted.

Validation of Handler Files

If a file is invalid, it is skipped, an error is logged, and the next configuration handler file (if present) is processed.

The following checks will be performed on all handler files to ensure correctness.

- Verification of the validity of the JSON file
- `handlers` is the top level key of the handler config file and a single `handlers` key should be specified

- The handler name cannot be empty
- Duplicate handlers are not allowed
- If `capabilities` are specified:
    - must be an array
    - must contain a non-empty `name`
    - must contain a non-empty `version`
- `commands` must be specified as part of a handler and the name of each command must be specified.
- The command `bin` must be specified and the path must point to a valid executable file.

Writing Commands

Command parameters are passed on the command line as `--key1=value1`, `--key2=value2`, etc. The order of these parameters is arbitrary and may change between invocations of the command.

Writing Status Commands

Status commands are special commands used in status reporting. If a handler implements a “status” command, Edge Agent will invoke this command to fetch the status of the handler while compiling the status report for the system. The result of the "status" command execution should be printed to standard output and should be a JSON document - a list of statuses for each capability exposed by the handler. Here is an example of the result of a "status" command:

```
[
  {
    "capabilityId": "Wifi1",
    "capabilityVersion": "1.2.1",
    "status": "On"
  },
  {
    "capabilityId": "Zigbee1",
    "capabilityVersion": "1.0.0",
    "status": "Stand-By"
  },
  {
    "capabilityId": "DCMotor1",
    "capabilityVersion": "2.0.0",
    "status": "Off"
```

```
  }
]
```

Please note that each object in the list must have these fields: `capabilityId`, `capabilityVersion` and `status`. However, the contents of these fields are not validated. It is the developer's responsibility to match capabilities against the capabilities the handler exposes, however, it is not mandatory.

Writing Package Deployment Commands

Parameters for a package deployment command are passed on the command line as `--key1=value1`, `--key2=value2`, etc. The package file name is passed on the command line as `--file=<full_path_to_the_package_file>`. The order of these parameters is arbitrary and may change between invocations of the command.

Task ID

Some commands require a task ID. Typically, this is done if a command must span a reboot (e.g., an operating system upgrade). If a command requires a task ID then `needs_task_id` must be set to "true" (the default is "false"). If `task_id` is true, the first parameter passed to the command is the task ID. It is NOT prefixed by any `--key=`.

Executable File

The "bin" variable is the executable file that corresponds to the command. It must be placed in the handler path specified in `handler.path` and be executable by `eauser`.

Triggering Commands

From Edge Manager with Custom Commands

Once the commands, handlers, and capabilities have been created, the next step is to trigger them remotely from Edge Manager. (See Adding a Custom Command *(on page)*.)

After (enrolling the device *(on page)*) into your Edge Manager instance, navigate to the **Commands** tab. From the **Action** drop down menu select **Add**.

- Enter a name and description that clearly indicates what the commmand does.
- Set the **COMMAND** and **HANDLER** fields to match what is set in the JSON, in this case 'date' and 'host' respectively.
- Select **Has output** in order to see the results of this command from the system.

You can then execute the command (see Executing Commands *(on page)*).

The new command will be included in the **Commands History** table. The status should cycle through **Pending**, to **In Progress**, and **Success**. The command's results can be downloaded from the **Output** column.

Locally with the Edge Core API

Commands can be triggered locally without enrolling to Edge Manager by using the Edge Core API available at `/var/run/edge-core/edge-core.sock`. Once loaded, the command will be available at `http://localhost/api/v1/host/commands/<handler>/<command>`.

An example using curl:

```
curl --unix-socket /var/run/edge-core/edge-core.sock \
     http://localhost/api/v1/host/commands/host/date \
     -X POST
```

Example success response:

```
{"output": "Service predix-edge-broker removed."}
```

Example error response:

```
{ "error_message": "Not Found", "status_code": 404}
```

Note:
The Edge Core API is not currently available on Ubuntu or Windows versions of Edge, as it is only required for use with the Predix Edge Technician Console (PETC).

Obtaining 'root' Permission When Required

By default, commands run under the user `eauser`. For some commands, you may require `root` permission. This section describes how to obtain `root` permission. This should be done only after exhausting all other possibilities (e.g., adding `eauser` to a privileged group).

Create Access Rules

There are two ways to create rules: INI-formatted `.pkla` files and javaScript-based `.rules` files.

- polkit 0.105 (and earlier) - *.pkla file
  - polkit 0.105 and earlier use `*.pkla` files to implement rules. This is the version of polkit used in all supported versions of Ubuntu and Debian.
  - Create a `*.pkla` file in `/etc/polkit-1/localauthority/50-local.d` to create a rule. For example,

```
[Allow eauser to run some-script as root]

Identity=unix-user:eauser

Action=com.system-builder.edge.some-script

ResultAny=yes
```

- polkit 0.106 (and later) - *.rules file
  - polkit 0.106 and later use `*.rules` files to implement rules. This is the version of polkit used in Edge OS.
  - Create an `80-sysbuilder-some-script.rules` file in `/etc/polkit-1/rules.d` to check for the `some-script action.id`.

```
polkit.addRule(function(action, subject) {

    if ((

          action.id == "com.system-builder.edge.some-script"

        )  && subject.user == "eauser") {

        return polkit.Result.YES;

    }

});
```

Enable Script to Run as a User

Add `pkexec` to the command's shebang line.

```
#!/usr/bin/pkexec /bin/sh
```

Declare Actions Managed by the Policy

Create a `com.system-builder.edge.some-policy.policy` file in `/usr/share/polkit-1/actions/`. For example:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE policyconfig PUBLIC

  "-//freedesktop//DTD PolicyKit Policy Configuration 1.0//EN"
```

```
  "http://www.freedesktop.org/standards/PolicyKit/1/policyconfig.dtd">
<policyconfig>


  <action id="com.system-builder.edge.pkexec.some-script">
    <description>allow running some-script with pkexec</description>
    <defaults>
      <allow_any>no</allow_any>
      <allow_inactive>no</allow_inactive>
      <allow_active>no</allow_active>
    </defaults>
    <annotate key="org.freedesktop.policykit.exec.path">/bin/sh</annotate>
    <annotate key="org.freedesktop.policykit.exec.argv1">/opt/edge-agent/some-script</annotate>
  </action>


</policyconfig>
```

Predix Edge Agent Release Notes

Edge Agent on Ubuntu Release Notes 24.04

Host OS Compatibility

- Ubuntu 22.04-LTS (Jammy Jellyfish)

Enhancements

The following Enhancements were implemented in Edge Agent Ubuntu 24.04.

AWS-signed Edge Device Certificates

Support has been added for Edge device certificates signed by AWS root certificate authority. This change applies to both new device enrollment and renewal when using Edge Manager for Edge device fleet management. No end-user action is required as a result of this change.

General Improvements

Internal stability fixes and component library updates.

Known Issues

This release contains the following known issues.

Debian Package Download

Installation from an apt repository has been deprecated. Please download the Debian package from https://dig-grid-artifactory.apps.ge.com/ui/native/predix-edge-agent-deb/pool/ and follow the Install From Downloaded Package *(on page xxviii)* instructions.

Debian Package Re-installation

When re-installing (not updating) the Edge Agent on Ubuntu debian package, remove the following users and groups: eauser, eagateway. For example:

- `sudo userdel -r eauser`
- `sudo groupdel eauser`

Edge Agent on Ubuntu Release Notes 23.11

Host OS Compatibility

- Ubuntu 22.04-LTS (Jammy)

Enhancements

There were no enhancements in this release.

Known Issues

This release contains the following known issue.

Debian Package Download

Installation from an apt repository is currently unavailable. Please download the Debian package from https://dig-grid-artifactory.apps.ge.com/ui/native/predix-edge-agent-deb/pool/ and follow the Install From Downloaded Packageinstructions.

Edge Agent on Ubuntu Release Notes 23.02

Host OS Compatibility

- Ubuntu 18.04-LTS

Enhancements

This release contains the following enhancements.

Volume Mount Restrictions for Docker Compose Files

Volume mount restrictions for Docker compose files have been modified to allow greater flexibility for the container application.

- Volume type should be `volume` or `tempfs`.
- The source of the volume must be one of:
  - A named volume.
  - Relative to the current working directory and below the current working directory (for example, `./src/volume/path` is valid; `./../../etc/passwd` and `/etc/passwd` is not).
- The destination of the volume must not be one of these reserved values:
  - `/config`
  - `/data`
  - `/edge-agent`

- `/edge-core`
- `/shared`

Image Name Requirements

Inside an Edge Application's `docker-compose.yml` file, images can now be pulled from a Docker Trusted Registry (DTR). It is recommended to reference such images by their hash. For example, it is preferred to use `image: ubuntu@sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d` instead of `image: ubuntu:22.04` because the exact image specified by the `22.04` tag (or any other tag) could change between deployments of your application. For more information see https://docs.docker.com/engine/reference/commandline/pull/#pull-an-image-by-digest-immutable-identifier.

If you choose to host your images in a different docker registry, they can be referenced in the docker-compose.yml file as well. For example:

```
my-app:

    image: myregistry.local:5000/testing/test-image
```

For more information see https://docs.docker.com/engine/reference/commandline/pull/#pull-from-a-different-registry. If the registry requires authentication, execute a `docker login` command on your Edge devices before attempting to deploy the app. For more information see Usage.

Known Issues

This release contains the following known issues.

Debian Package Download

Installation from an apt repository is currently unavailable. Please download the Debian package from https://dig-grid-artifactory.apps.ge.com/ui/native/predix-edge-agent-deb/pool/ and follow the Install From Downloaded Packageinstructions.

Edge Agent Release Notes 20.09.0

Edge Agent on Ubuntu

We have made available a package for Edge Agent on Ubuntu 18.04 that will allow you to:

- Manage devices using Edge Manager
- Install and manage Edge applications
- Manage Ubuntu packages using 'apt' from Edge Manager

Note:
Edge Agent on Ubuntu 18.04 (`predix-edge-agent_20.09.0-1601066661-3d2e9d53_amd64.deb`) supports only libprotobuf10, even though it can be installed with later versions of libprotobuf (e.g., libprotobuf17).

See Edge Agent on Ubuntu.

Predix Edge Agent Release Notes 2.4.0

Enhancements

This release has the following enhancements:

Sends Static Device Information to Edge Manager

Static device information, such as memory total bytes, number of cores, disk name and size, and the device's physical model name can now be sent to Edge Manager.

Set Polling Forces Synchronization

The set polling command now immediately forces a synchronization.

Additional Event Log Functionality

The following event log functionality has been added:

- Ability to list event logs from Edge Manager
- Ability to retrieve event logs from Edge Manager
- Log entries created for events such as:
  - Edge Manager communication errors when polling
  - Resolving a situation where the MQTT password is corrupted during a power cycle

Known Issues

This release has the following known issues:

Docker Apps Flooding Logs

A Docker application that floods the logs can cause system performance to degrade, resulting in the Docker applications restarting.

Serial Port Bootlog

A qemu- or vmware-based OS will output the bootlog out of the serial ports. This can cause problems if they are connected to physical hardware.

Secure Command Channel

An abrupt loss of power while deploying a secure command channel-enabled application may result in an inability to manage applications that leverage the secure command channel.

Device Clock Modifications

If a device's clock is modified to a time prior to when it was first initialized, all deployed applications will be stopped and all system applications will run after the device reboots.

Application Configurations

Applications will not be able to read application configuration files created without read permissions. Ensure that configuration files have read permissions prior to zipping them in preparation for uploading them to Edge Manager or PETC.

Predix Edge Agent Release Notes 2.3.3

Bug Fixes

This release contains the following bug fixes:

Memory Leak

Fixed a memory leak relating to Edge application status that would eventually cause the docker daemon to be terminated by the out-of-memory killer.

Device Unresponsive From Edge Manager

Fixed an issue that could result in an Edge device becoming unresponsive to Edge Manager if a package download was extremely slow or failed due to certain network conditions.

Known Issues

This release has the following known issues:

Docker Apps Flooding Logs

A Docker application that floods the logs can cause system performance to degrade, resulting in the Docker applications restarting.

Serial Port Bootlog

A qemu- or vmware-based OS will output the bootlog out of the serial ports. This can cause problems if they are connected to physical hardware.

Secure Command Channel

An abrupt loss of power while deploying a secure command channel-enabled application may result in an inability to manage applications that leverage the secure command channel.

Device Clock Modifications

If a device's clock is modified to a time prior to when it was first initialized, all deployed applications will be stopped and all system applications will run after the device reboots.

Application Configurations

Applications will not be able to read application configuration files created without read permissions. Ensure that configuration files have read permissions prior to zipping them in preparation for uploading them to Edge Manager or PETC.

Predix Edge Agent Release Notes 2.3.1

Security Issues

This is an out of band maintenance release for security issues identified in Predix Edge Agent, versions 2.3.0 and prior. It is recommended you apply this fix as soon as possible, especially if you are running Predix Edge in a production environment.

- Fixed an issue that could result in local privilege escalation for an attacker with write access to /tmp.
- Fixed an issue where a signed SSH key could bypass validation.
- Fixed an issue where a Predix Edge application could bypass application signing enforcement.
- Fixed an issue where a Predix Edge application could bypass validation.

Predix Edge Agent Release Notes 2.3.0

These are the new features and known and resolved issues for Predix Edge Agent, version 2.3.0.

I

New Features

This release contains the following new features:

General

- Ability to manage SSH keys, allowing support staff to enable, disable, and list deployed keys. These keys must be signed by the GED support team.
- Applications can now access serial ports exposed by the operating system.
- Ability for a user to erase all data in the device. This will reset the device to the currently installed operating system default.

> **Note:**
> This will not perform a secure delete.

- The `/api/v1/host/os` endpoint now includes the `machine_type`, which indicates whether the device is an embedded or virtual device.

Known Issues

This release has the following known issues:

Docker Apps Flooding Logs

A Docker application that floods the logs can cause system performance to degrade, resulting in the Docker applications restarting.

Serial Port Bootlog

A qemu- or vmware-based OS will output the bootlog out of the serial ports. This can cause problems if they are connected to physical hardware.

Secure Command Channel

An abrupt loss of power while deploying a secure command channel-enabled application may result in an inability to manage applications that leverage the secure command channel.

Device Clock Modifications

If a device's clock is modified to a time prior to when it was first initialized, all deployed applications will be stopped and all system applications will run after the device reboots.

Application Configurations

Applications will not be able to read application configuration files created without read permissions. Ensure that configuration files have read permissions prior to zipping them in preparation for uploading them to Edge Manager or PETC.

Predix Edge Agent Release Notes 2.2.0

These are the new features and known and resolved issues for Predix Edge Agent, version 2.2.0.

New Features

This release contains the following new features:

General

- Ability to execute commands from Edge Manager into an application.
- Ability for an application to expose capabilities and related status to Edge Manager.
- Ability to implement an application that supports the `Predix.Edge.AnalyticEngine` capability.
- Ability to execute the following commands from Edge Manager:
  - **Run Top**
  - **Run ifconfig**
  - **Set Polling Interval**

Known Issues

This release has the following known issues:

Docker Apps Flooding Logs

A Docker application that floods the logs can cause system performance to degrade, resulting in the Docker applications restarting.

Application Data Directory Ownership

If ownership of the `application/data` directory is modified by the application, subsequent updates or redeployments of the application will fail.

- Workaround: If you must change the owner of a directory, leverage a sub-directory under the `/data` directory, rather than the root directory.
- If the workaround was not implemented and attempts to update or redeploy the application fail, you must delete and re-add the application to update it. This will result in a loss of the content in the `/data` and `/config` directories.

Application Upgrade Failure

An application that creates a volume mount at `/data` in the Docker build file can also cause an upgrade of the application to fail.

- Workaround: Remove the `/data` or `/config` volume mounts from the Docker build file. These mounts will automatically be created at deployment..
- If the workaround was not implemented and attempts to update or redeploy the application fail, you must delete and re-add the application to update it. This will result in a loss of the content in the `/data` and `/config` directories.

Extended Power Off Period

Docker will not properly start if a system with a working clock has been powered off for more than three months. Existing applications will be stopped and will not be able to be re-started. A log entry will indicate that "swarm component could not be started" with a x509 certificate expired error if the device is in this state.

To avoid this issue, do not power off a device for more than three months. If you encounter this issue, contact technical support.

Upgrade Notes

- `edge-agent-runner` is now a reserved image name; any application that utilizes an image named ‘edge-agent-runner’ will be rejected during deployment.
- `edgeAgent/` is now a reserved topic in the MQTT broker; applications utilizing topics under this hierarchy may no longer work.
- `edgeApp/` is now a reserved topic in the MQTT broker; applications utilizing topics under this hierarchy may no longer work.

Predix Edge Agent Release Notes 2.1.0

These are the new features and known and resolved issues for Predix Edge Agent, version 2.1.0.

New Features

This release contains the following new features:

Raspberry Pi

Reference implementation of Edge Agent on Raspberry Pi 3 B+.

Predix Certificate

Automatic renewal of the Predix certificate.

Resolved Issues

The following issues were resolved in this release:

Updating Proxy Settings

When updating the proxy settings in previous versions, previously deployed applications' environment variables would not be updated to reflect the change. This behaviour has been modified and the applications will now automatically redeploy in the subsequent reboot sequence, which will refresh the environment variables.

Inaccurate Timestamp After Reboot

Edge Manager-based commands that required a reboot would previously report back inaccurate start times. This has been resolved.

Known Issues

This release has the following known issues:

Docker Apps Flooding Logs

A Docker application that floods the logs can cause system performance to degrade, resulting in the Docker applications restarting.

Redeploying/Starting Existing Applications

When redeploying or starting an existing application, the operation will erroneously return a warning relating to the possibility of running different versions of the image.